AFRL-IF-RS-TR-2004-39
Final Technical Report
February 2004

ROBUST AGENT-BASED SYSTEMS INCORPORATING TEAMS OF COMMUNICATING AGENTS

Oregon Health & Science University

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. J383

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

STINFO FINAL REPORT

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2004-39 has been reviewed and is approved for publication.

APPROVED: /s/
DANIEL E. DASKIEWICH
Project Engineer

FOR THE DIRECTOR: /s/
JAMES A. COLLINS, Acting Chief
Information Technology Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	FEBRUARY 2004	**FINAL** **Apr 98 – May 03**

4. TITLE AND SUBTITLE

ROBUST AGENT-BASED SYSTEMS INCORPORATING TEAMS OF COMMUNICATING AGENTS

5. FUNDING NUMBERS

G - F30602-98-2-0098
PE - 63760E
PR - AGEN
TA - T0
WU - 02

6. AUTHOR(S)

Philip R. Cohen

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Oregon Health & Science University
2000 MW Walker Road
Beaverton OR 97291-1000

8. PERFORMING ORGANIZATION REPORT NUMBER

N/A

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Defense Advanced Research Projects Agency
3701 North Fairfax Drive
Arlington VA 22203-1714

AFRL/IFTB
525 Brooks Road
Rome NY 13441-4505

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

AFRL-IF-RS-TR-2004-39

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: Daniel E. Daskiewich/IFTB/(315) 330-7731 Daniel.Daskiewich@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 Words)***

This report describes OGI's contribution to agent based systems design and prototyping. The project had two major thrusts: 1) the development of the Adaptive Agent Architecture (A3), and 2) the development of a second generation agent communication language (AgentTalk). The A3 architecture supports human participation as agents, dynamic formation of agent teams, reorganization of teams based on network loads, and offers libraries for legacy code to be able to participate in agent based systems. AgentTalk offers a small set of primitive communication actions, operators with which to build more complex communication activities, provably correct dialogue protocols, and is a candidate for international standardization.

14. SUBJECT TERMS

Software Agents Multi-modal Interaction Fault Tolerance

15. NUMBER OF PAGES 29

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Table of Contents

Table of Figures

Abstract

This project addressed three distinct topics: 1) Building a fault-tolerant multi-agent systems framework inspired by the theory of joint intentions, 2) developing a multi-agent systems framework that directly executes a joint intention specification, 3) exploring the use of multimodal interaction in field and mobile environments. Topic 1 occupied the first two years of this project, while topics 2 and later 3 have been a more recent undertaking. Since topics 1 and 2 have been written about before, we only summarize them here, providing pointers to the relevant papers, and also include an appendix that provides a paper in preparation. Topic 3 is new and has not been written about before, so we present the basic results here. A paper will be submitted to an upcoming user-interfaces conference.

1 FAULT-TOLERANT MULTI-AGENT SYSTEMS

We observed that multi-agent systems are prone to failures typical of any distributed system. Agents and resources may become unavailable due to machine crashes, communication breakdowns, process failures, and numerous other hardware and software failures. Most of the work done in fault handling in multi-agent systems deals with detection and recovery from faults such as state-inconsistencies, relying on the traditional techniques for recovering from other distributed systems failure. However, the traditional fault-tolerance techniques are designed for specific situations and they require special infrastructural support. We showed that fault-tolerance techniques can be readily implemented using generic agents with minimal or no modification to the agent infrastructure. We demonstrated that theories from multi-agent systems literature (eg., joint intention theory [2]) can be effectively combined with basic fault-tolerance principles to design robust multi-agent systems. In particular, we showed that (1) teamwork can be used to create a robust brokered architecture that will recover a multi-agent system from broker failures without incurring undue overheads, (2) teamwork can also be used to guarantee a specified number of brokers in a large multi-agent system, and (3) agent autonomy can be used to prevent thrashing and guarantee acceptable levels of quality of service by an agent. To validate our approach, we developed and experimented with the Adaptive Agent Architecture (AAA). AAA is a facilitated and peer-to-peer multi-agent system architecture that offers teams of facilitators for fault-tolerance. It is backwards compatible with the Open Agent Architecture (OAA) [1, 6], and is used as an interoperation framework both within our QuickSet multimodal system, and also across numerous applications. Indeed, AAA was instrumental in tying together the facilitators that supported the first Control of Agent Based system (CoABS) Technology Integration Experiment (OAA and the CoABS Grid). The research on AAA contributed to the teamwork theory itself by way of a formal characterization of persistent teams and maintenance goals.

1.1 Adaptive Agent Architecture (AAA)

The AAA is a multi-brokered multi-agent system architecture that provides an infrastructure for building fault-tolerant brokered multi-agent systems. As mentioned earlier, it interoperates with the Open Agent Architecture and it is currently used in multi-agent systems such as Quickset that place heavy demand on inter-agent communication and yet provides acceptable real time response. The AAA agent library has been developed in Java and it provides an agent shell for developing AAA agents. The library also provides a facilitator agent that serves both as a broker and a matchmaker. The AAA facilitators can be interconnected to form arbitrary topologies, thus allowing fast dynamic federation of the AAA agent communities that are not limited to hierarchical multi-blackboard architectures. The agent library supports both facilitated and direct inter-agent communication. The AAA agents advertise their capabilities as well as an address for connection requests with a broker during registration. In case of direct inter-agent communication, the agents use the facilitator as a matchmaker to find other capable agents and contact them directly. This is particularly useful when communicating large volumes of data with large message size (such as ink data) that is typical in multimodal interface agents. TCP/IP is used for network transport and the TCP mechanisms and timeouts are used for detection of connection failures. The facilitators as well as other agents can dynamically enter and leave AAA-based multi-agent systems. The AAA brokers form a team for the purpose of fault-tolerance and they share knowledge about who is connected to whom with the team members.

The AAA library uses Horn-clause for knowledge representation, and provides a prolog-oriented term hierarchy to automate parsing and to provide full unification of messages. It provides a thoroughly streamlined Application Programming Interface (API), reducing the amount of code needed to implement an agent, and the API is based around speech acts (e.g. request and inform). Special software engineering features allow automatic conversion to and from java objects and their representation in the particular agent communication

language being used. The library includes a persistent BlackBoard data structure that is used to write persistent agents. Further, the AAA facilitators use this BlackBoard internally and so they can be made persistent on demand. The AAA-based multi-agent systems interoperate with the Defense Advanced Research Projects Agency (DARPA) CoABS Grid wherein AAA Facilitators treat the JINI based grid as an external matchmaking resource. Using AAA with the Grid in this fashion eliminates the use of setup files because the AAA agents and the facilitators find each other through the Grid.

1.2 Theoretical Results

The fault-tolerant behavior of AAA-based multi-agent systems is a consequence of these teamwork specifications. We formalized a notion of team commitment that allows for dynamic but persistent teams whose members can change with time and also formally defined restorative maintenance goals that enable a team to exist beyond one-time joint persistent goals. These two notions of persistent teams enable an AAA-based multi-agent system (1) to recover from broker failures that arise from problems such as machine crashes, communication breakdown, and death of a broker process, and (2) to maintain a specified minimum number of functional brokers in the system even when some of the brokers become inaccessible. We briefly discuss the two teamwork behaviors next and the formal details can be found in [5].

1.2.1 Persistent Teams

Team activity is explained in terms of the theory of joint intentions [2]. This theory characterizes an agent's behavior in a team in terms of its internal state described in modal logic, linear time temporal logic, and dynamic logic of action. This earlier work characterized a team in terms of joint commitment between the individuals that originally constituted the team. As such, the joint commitment ceases to exist when either of the team members ceases to exist. On the other hand, teams in the real world may be one-time teams that are disbanded after the team goal has been achieved, or they may be persistent teams that continue to exist even when the team members change. For example, the Boston Red Sox are a team even if all the players are traded or it is sold to new owners. Persistent teams are especially desirable from a fault-tolerance perspective because agents that fail will generally be replaced by other agents during the recovery process. To support fault-tolerance of AAA facilitator teams, we enhanced the previous teamwork theory such that a team can exist independently of the identity of its members. The AAA research introduced a notion of team commitment wherein the agents are committed towards "whoever" are members of the team at any time, thus enabling the team to continue even when the team membership changes dynamically. The notions of team commitment and team intentions were redefined with respect to the team as an entity. This allows a team having these commitments to continue as long as there is no mutual belief about the completion, impossibility or irrelevance of the team goal even if some of the team members leave and new members join the team. The achievement of mutual belief among team members requires group communication that was another of our research efforts under the CoABS program and is described in section 2.

The design of the AAA facilitator implements the specification of teamwork that follows from the following mission statement: "whenever an agent registers with the facilitator team, the brokers have a team intention of connecting with that agent, if it ever disconnects, as long as it remains registered with the team." In other words, the facilitators jointly intend that:

For each agent A
While A is registered, if A is not connected, connect to A

Using the mission statements, along with other logical properties of the AAA, we logically establish the commitments of the brokers in the team and predict who communicates what to whom in order to satisfy the above joint intention. Essentially, AAA facilitators end up having a joint commitment to service agents

registered with any of the broker teammates with the assumption that brokers can initiate connection with stranded agents when a broker teammate dies. These commitments result in fault tolerant behavior when the brokers act rationally and take appropriate actions to honor them. These commitments are in addition to any brokering commitments that the broker team may have. It is to be noted that the teamwork theory is not directly implemented in AAA brokers, the theory (as used in the above mission statement) provides the software design specification that is then implemented as part of the facilitator code.

1.2.2 Maintenance Goals

The earlier characterization of teamwork in terms of joint commitment results in the team being dissolved once the jointly committed goal is mutually believed to be either achieved or impossible or irrelevant. However, fault-tolerance is a continuous phenomenon and therefore requires that teams be not just one-shot teams. We proposed a characterization of team commitment for maintenance goals in order to enable teams that continue beyond one-time jointly committed goals.

Maintenance goals can be restorative or preventive. We were concerned with only the restorative maintenance goal for the AAA fault-tolerance. We say that an agent has a (restorative) maintenance goal if the following is true of the agent: if the agent does not believe that the proposition to be maintained is true, it will adopt the goal that the proposition be eventually true. The maintenance goal is persistent if this fact remains true of the agent at least until the agent either believes that it is impossible to maintain that proposition or that the maintenance goal is irrelevant. Similarly, a team of agents has a team maintenance goal to maintain a proposition if whenever the team comes to mutually believe that the proposition is not true, it adopts a joint commitment to achieve that proposition. This new jointly committed goal is no longer valid once the required proposition is achieved and mutually believed, but the original maintenance commitment remains valid and the team continues to exist at least until the team mutually believes the impossibility or the irrelevance of maintaining that proposition.

The design of the AAA facilitator also implements the specification of teamwork that follows from the following mission statement: “the AAA facilitator team has a team maintenance goal of having at least N facilitators in the team at all times where N is specified during the team formation.” In other words, the facilitator team has the joint restorative maintenance goal that says:

The facilitator team is committed to P staying true,
but if it ever becomes false, the team is committed to re-achieving it
where P = #brokers >=minimum.

Using logical properties of AAA along with this mission statement, we have shown that it results in reconstituting a persistent broker team after broker failure by jointly intending to maintain a minimum number of facilitators with the assumption that agents can start new brokers on different machines within security constraints, and without additional permissions. Just as with the first mission statement, the above mission statement serves as a software design specification for the AAA facilitators.

1.3 Experimental results

AAA-based multi-agent systems exhibited the robustness from facilitator failures once the teamwork specifications discussed above were implemented in the AAA facilitators. Thereafter, we designed and ran a series of experiments to empirically test our hypothesis that it was possible to achieve such fault-tolerance in multi-agent systems in a networked as well as standalone environment without appreciable interference with the normal operation of the system. The results of the experiment are discussed next for the networked

environment. Two agents names as “Distance Agent” and “Client Agent” communicated with each other via messages that were relayed back and forth through an interconnection of facilitators acting as a team. The agents and facilitators were distributed over a network of four machines and the facilitator processes were killed randomly during the experiment. The details of this experiment are discussed in [4].

1.3.1 Effect of Recovery on Response Time

The experiment in Figure 1 was started with six fully connected facilitators and the facilitators were killed one by one at random. Three different machines were running two facilitators each and the client and distance agents were running on the fourth machine. The stair-step plot shows the number of brokers present at any time. The agents mentioned near the peaks are the agents that were connected to the broker that was killed.

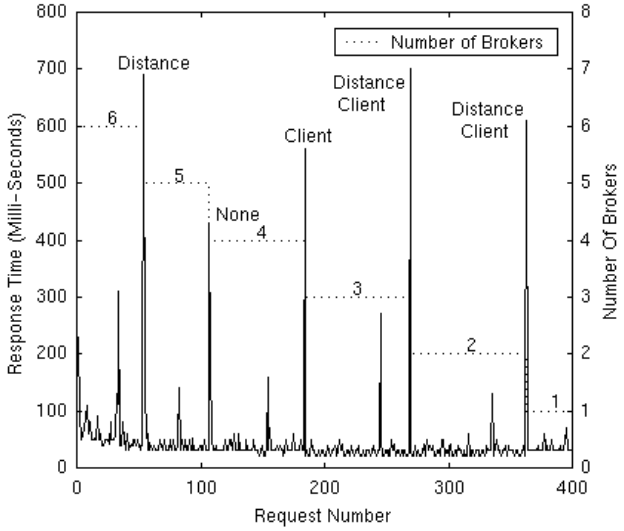

Figure 1: Effect on response time as facilitators fail

The plot in Figure 1 clearly shows that *the AAA recovery scheme enables the system to function, despite facilitator failures, as long as there is at least one facilitator left in the system.* The peak response times are for the requests that were in progress when the broker got killed. The peak value depends on (1) the load on different machines, (2) latencies of the devices on different machines and the network latency, (3) operating system latencies such as thread scheduling, and (4) the state of the request when the broker failed. As such, no inference can be made about the pattern of peak response times except that they happened to be within an order of magnitude from random spikes in this experiment. The smaller spikes in response times preceding the peaks caused by broker failures is probably a result of disturbing the system (and hence, enhancing the random effects) in the process of manually killing the brokers.

1.3.2 Overheads of Using Teamwork

The establishment of joint persistent goals for teamwork requires communication overheads. However, the implementation of teamwork described above comes into play only when there is some transition in the system such as facilitators or agents being added or removed. As such, we would expect that there should be no teamwork overhead in the steady state. Figure 2 shows two plots, one with the normal recoverable system, and the other with the teamwork code disabled. For each number of facilitators, the response time was collected for 100 requests and the maximum, the minimum and the mean have been plotted.

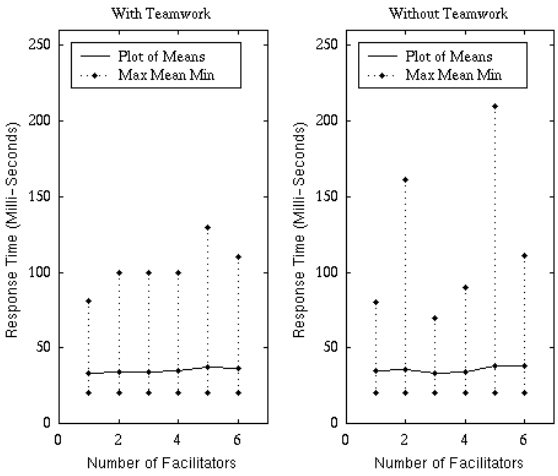

Figure 2: Mean response times with and without teamwork

We used pair wise t-tests to determine whether or not the difference between the mean response time with teamwork behavior enabled and the mean response time with teamwork behavior disabled was statistically significant. The t-tests were repeated after rejecting the outliers given by a box plot. The p-values were very high (about an order of magnitude greater than the significance level 0.05) in both cases, thereby giving no indication of any statistically significant difference in the mean response time in the steady state as a result of using teamwork.

1.3.3 Effect of Transitions on Response Time

Facilitator-team reorganization does affect the response times of the requests in progress at the time of the reorganization. The effect of adding additional facilitators to a working system is expected to be similar to that when facilitators are killed (assuming that the effects of the CPU getting overloaded due to additional processes and the disturbance due to process creation are accounted for).

When an agent enters or leaves the multi-agent system, a message is sent to all the facilitator teammates as prescribed teamwork theory when applied to the mission statements of the AAA facilitators. However, this message generates too little traffic to appreciably affect the ongoing agent conversations. The experiment in Figure 3 consisted of four interconnected facilitators running on two different machines. The client and distance agent were running on a third machine. Seven additional agents were started on the fourth (and the fastest) machine one by one every two seconds and then killed one by one every two seconds. These additional agents registered with one of the facilitators but did not participate in the interaction between the client agent and the distance agent. The starting and killing of these agents was automated to avoid disturbing the system by manual interaction. The system was allowed to run until completion.

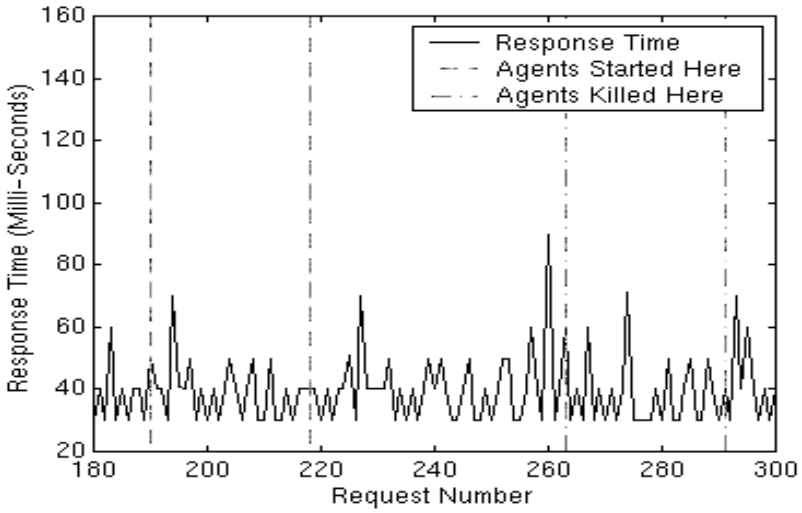

Figure 3: Response time when agents leave the system

For clarity of the figure, we show here a portion of the entire plot (the rest of the plot exhibits the same pattern as in this figure). We observe that there is no appreciable and consistent peak in response time whenever an agent was started or killed. Moreover, small peaks if any, are within the bounds of random effects (the peaks in response time that we see when no agent was started or killed is due to random effects). This result met our expectation that agents entering or leaving an AAA-based multi-agent system do not cause the teamwork implementation to create a noticeable disturbance in the application.

1.4 CoABS Grid

The AAA facilitators were made "grid-aware" and were one of the components on the CoABS grid. AAA facilitators treat the Grid as an external matchmaker, and communication with the Grid is through a matchmaker interface *within* the AAA facilitators. Each facilitator registers its capabilities with the Grid, and looks for other facilitators on the Grid when it comes up and registers with them, if not already registered, forming the aforementioned fault-tolerant facilitator team. Any additional facilitators spawned by AAA agents, as part of the fault-tolerant process, also register with the Grid as well as with other facilitators they discover on the Grid.

The Quickset-Multimodal Map Interoperator was another AAA agent on the CoABS Grid. Along with the AAA facilitators, this Interoperator agent enabled collaboration between agent communities of two different research groups (Stanford Research Institute's multi-modal map and Oregon Graduate Institute's Quickset), and was an integral part of the first CoABS technology integration experiment. Various Quickset agents also became accessible from the CoABS Grid as a result of the AAA facilitator being on the Grid, and these include the following potentially useful agents:

–Gesture Recognizer Agent (for Military Symbology)
–Natural Language Processor Agent
–Speech Recognizer Agent
–Handwriting Recognizer Agent
–Multi-modal Integrator Agent
–Quickset Collaborative User Interface Agent (for Sharing Ink)

–GateKeeper Agent (Persistent Store for Military Symbology)

The CoABS program had a "24/7 CoABS Grid" wherein the Grid software run locally by the CoABS researchers across the country interconnected to form a giant Grid, and it was in operation 24 hours a day, 7 days a week. The services of the above mentioned Quickset agents were made available to other CoABS researchers via this 24/7 CoABS Grid.

1.5 Conclusion

The AAA is a result of our CoABS research effort on agent architectures that were robust to failures of middle agents such as the facilitators and brokers. The AAA library, including the fault-tolerant facilitators has been available for download by the DARPA CoABS team and the research community. It is currently being used by a number of government and university research labs as well as companies for various agent-based multi-modal applications. Further research details about the AAA can be found in the following publications.

1.6 Publications:

Kumar, Sanjeev; Cohen, Philip R. Towards a Fault-Tolerant Multi-Agent System Architecture. In Proceedings of *The Fourth International Conference on Autonomous Agents (Agents 2000)*, Barcelona, Spain, June 3-7, 2000.

Kumar, Sanjeev; Cohen, Philip R.; Levesque, Hector J. The Adaptive Agent Architecture: Achieving Fault-Tolerance Using Persistent Broker Teams. In Proceedings of *The Fourth International Conference on Multi-Agent Systems (ICMAS 2000)*, Boston MA, USA, July 7-12, 2000.

2 DEVELOPMENT OF STAPLE

We have been working on a multi-agent programming language called STAPLE (*Social and Team Agents Programming Language*) with built in support for teamwork and multi-agent conversations. The motivation behind STAPLE has been to design an interpreter that can directly execute the logical specification of fault-tolerance in AAA facilitators (i.e. the mission statements) and get the resulting robust behavior without having to implement that specification. The logical specification can then be modified to get different behavior without having to implement the modified specification. The notion of teamwork and multi-agent conversation is at the core of the underlying theory behind STAPLE, and STAPLE agents reason about their joint intentions as well as about the ongoing conversations. Unlike the commonly used communication languages KQML (Knowledge Query and Manipulation Language) and FIPA (Foundation for Intelligent Physical Agents), the communication in STAPLE is based on a provably correct formal semantics of multi-agent conversations that was also developed as part of this DARPA CoABS program. The main contribution of STAPLE to the teamwork research is in taking the concept of directly executing team specifications to the next step by offering interpretation of agent specifications in a logic that is used for formal specification of the joint intention theory. Here we briefly discuss STAPLE as well as our research on multi-agent communication.

2.1 Overview of the Language

STAPLE enables programming multi-agent systems by directly executing the specification of agents in a subset of modal logic, dynamic logic of actions, and temporal logic along with abstractions from the joint intention theory as well as from a formal semantics of multi-agent conversations based on the joint intention theory. As such, STAPLE is a domain independent agent programming language that is formally connected with a logical theory of agency, and whose constructs, including the communication primitives, have a well-founded formal semantics. The benefits of this approach include the ability to modify agent and team behaviors just by modifying a single sentence in the logical language, and a potential for verification, for instance, one may be able to predict a team behavior offline using the logical specification of agents involved and verify it by running the actual system.

The formal language behind STAPLE is a modal language with the usual connectives of a first order language with equality, as well as operators for propositional attitudes and event sequences. (BEL x p) and (GOAL x *p*) say that *p* follows from x's beliefs or choices respectively. (HAPPENS *a*) and (DONE *a*) say that a sequence of actions described by the action expression *a* will happen next or has just happened, respectively. Temporal properties are expressed in a linear time temporal logic. $\Diamond p$ says that the proposition *p* will eventually be true, and $\Box p$ says that *p* will always be true. Letters such as τ are used to represent groups. (HAPPENS τ *a*) and (DONE τ *a*) specify the group τ as actor for the action sequence *a* without regard to which group members participate in the group action. An action expression is built from variables ranging over sequences of events using constructs of dynamic logic: a;b is action composition, a|b is non-deterministic choice, a||b is concurrent action, *p*? is a test action, and a* is indefinitely many repetitions. Complex action expressions are composed using these dynamic logic constructs. Mutual belief is defined in terms of unilateral mutual belief. Persistent goal (PGOAL) formalizes the notion of individual commitment and intention (INTEND) to do an action is defined to be a commitment to do the action knowingly. Team commitment (TPG) and team intention (TI) are similarly defined for a team of agents. The model theory of this language is based on a possible-worlds semantics. Worlds are modeled as a linear sequence of primitive event types and the possible worlds can be related to each other via belief and goal accessibility relations. Details of this modal language and its semantics can be found in the pointed papers.

Beliefs, goals, commitments, and intentions are represented explicitly in STAPLE. Actions are required to have a logical representation that can be used for reasoning, and plans as well as conversation protocols are treated as complex action expressions consisting of action sequences, non-deterministic OR, concurrent actions, repetitions, and test actions as in the original modal language. The axioms of rational behavior are specified as rules, and agent programmers can override the default rules. STAPLE agents can have multiple simultaneous commitments and intentions, and a notion of importance is used to order everything from commitments and intentions to plans and rules. The syntax of STAPLE is presently an extension of the usual Prolog syntax with the exception that certain constructs such as primitive actions and plans can be written in both Prolog and Java. Most of the STAPLE interpreter, including a multi-threaded Prolog subsystem, has been implemented in Java but a few components such as the belief reasoner and the default rules have been written in Prolog.

2.2 Implementation

STAPLE interpreter needs logical reasoning capabilities along with the ability to handle procedural tasks such as control flow and stack manipulation. Logic programming languages such as Prolog are good for logical reasoning but procedural tasks can quickly get quite complex and unwieldy in such languages. It was quite clear from our previous experience with STAPLE that completely implementing the STAPLE interpreter in Prolog was not a viable option. This is because a large portion of the interpreter dealt with procedural control,

and the commercially available multi-threaded Prolog implementations were too buggy and quickly broke down when used with STAPLE. Similarly, imperative languages such as Java are good for procedural tasks but are ill-suited for logical reasoning. As such, we take a hybrid approach for the current STAPLE development by choosing to use both Prolog and Java, and using each language for tasks that they do best. Java is used for procedural control and Prolog is used for logical reasoning. We implemented a multi-threaded Prolog library in Java, thereby, closely integrating these two disparate languages for use in the development of the STAPLE interpreter.

As in any interpreted language, a STAPLE program is first parsed and the appropriate data-structures are initialized. The agent's beliefs, actions, plans, and rules are placed in the appropriate databases, its commitments and intentions initialize the stacks that are used to interpret and keep track of the progress of commitments and intentions, and any observers (abstraction of sensors), and actuators (abstraction for effectors) required by the agent at startup are activated. Thereafter, the main interpreter loop is started and execution of the agent proceeds so as to achieve its commitments and intentions. Figure 4 shows the main components of the STAPLE interpreter that make this execution of STAPLE agents possible.

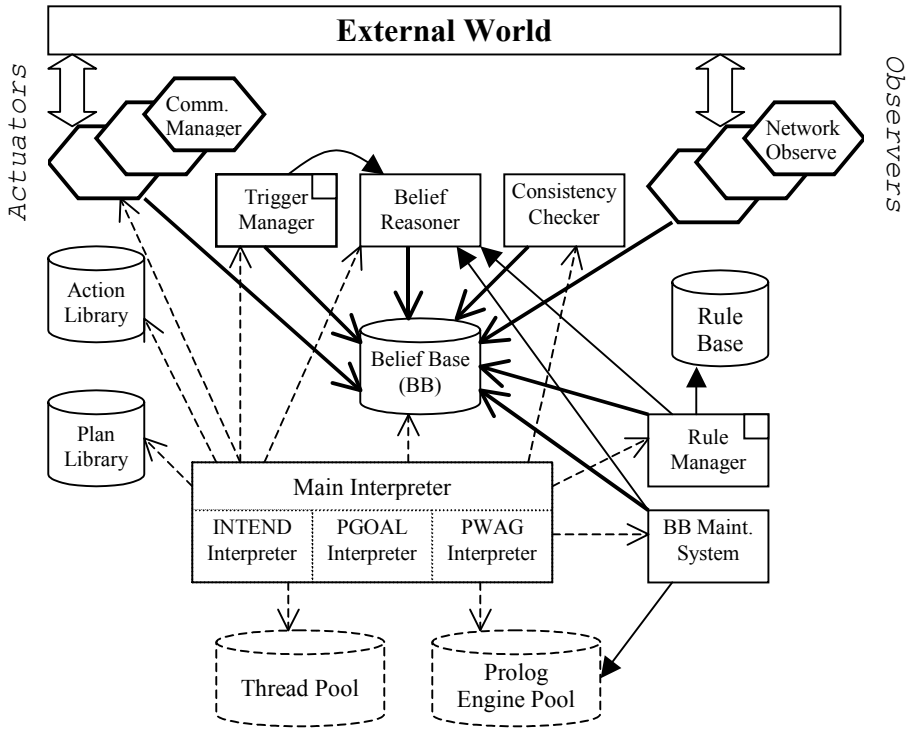

Figure 4: Main Components of STAPLE Interpreter

The directed arrows in Figure 4 show the dependency of the components, thereby indicating which components invoke or use which other components. The type of the arrows just indicates visual grouping of components. For instance, the bold arrows show the components that depend on the belief base, and the dotted arrows show the components that the main as well as the various modal interpreters depend on for their functioning. The components shown with notched corners (trigger manager and rule manager) have their own dedicated Prolog engines, and all other components that need access to a Prolog engine get it from a pool of reusable Prolog engines. All prolog engines share the same thread-safe knowledge base that has a very fine-grained locking granularity to allow highly concurrent access. The belief reasoner and the belief base maintenance systems are written entirely in Prolog, the consistency checker and the rule manager are written partly in Java and partly in Prolog, and everything else is written entirely in Java. The detailed working of each component in Figure 4 can be found in the appendix.

2.3 Multi-agent Conversation

A major part of our CoABS effort was directed towards developing an agent communication language with a well-defined formal semantics. Towards this end, we developed formal semantics for group communication, and conversation protocols.

2.3.1 Group Communication

Artificial as well as human agents not only interact with individual agents, but they also need to communicate with groups of agents. Moreover, in open multi-agent systems, where agents come and go dynamically, it will become ever more prevalent that agents will not know exactly to whom they are sending information or from whom they are requesting aid. In fact, a large number of distributed software systems inevitably use some incarnation of broadcasting and multicasting. However, there does not exist any formal semantics of multicasting and broadcasting in literature. Even the major agent communication languages have either no provision or no well-defined semantics for group communication. For instance, in the FIPA ACL (Agent Communication Language), the only way to inform a set of agents is to inform them individually, one at a time. Furthermore, semantics of the FIPA communicative acts imposes the precondition that the sender has certain beliefs about the mental state of the (known) addressee. Consequently, there is no way to send messages to unknown agents as in broadcast communication. DARPA's earlier agent communication language, KQML, does offer several primitives, such as broadcast and recruit-all, that have group flavor but these primitives are merely shorthand for a request to do a series of other communicative acts. Proper semantics cannot be given to group requests such as "One of you, please, get me a slice of that pie." We may safely conclude that support for group communication in the widely used agent communication languages does not exist.

Group communication is not just about sending a message to a large number of agents at the same time. As mentioned earlier, sometimes the sender does not know the specific recipients of a message. A person who posts the notice "Beware of dogs" may not know who will read that message. So the semantics of a communication language should allow for intentions with respect to "whoever gets this message," while allowing for constraints on the intended recipients and identification of this constraint for correct illocutionary effect. Furthermore, the intended actor for a communication may be a subset of the recipients or a completely different set. By sending an email to the CSE101 mailing list requesting Becker to take the attendance in the next class, the instructor not only made a request to Becker to take attendance but also let the whole class know that she requested Becker to do it. Senders need not only be individuals but can also be groups. An invitation card from John and Betty is actually a request to attend from "them". Individuals may be viewed as singleton groups. Therefore, the same communication primitives should work both for individual and group communication. We believe that any general-purpose agent communication language should be able to deal with these aspects of communication, and we are glad to report that our CoABS effort resulted in a well-defined formal semantics of group communication that remedies the aforementioned problems. Details of this semantics along with a set of desirable properties for any group communication language appear in (Kumar, et. al., 2000).

In fact, our work on group communication helped pinpoint certain errors and suggest fixes in the implementation of "send message to group" communication primitives in the CoABS Grid.

2.3.2 Conversation Protocols

Conversation protocols are traditionally specified as finite state machines in which the transition arcs specify the communicative actions to be used by the various agents involved in a conversation. Protocols are executed by performing these communicative actions and therefore, the communicative actions have come to be

regarded as the central concept around which analyses of protocols are based. However, we believe that it is the states and not the state transitions that are key to the correctness and completeness of a protocol. We propose a landmark-based approach for formal analysis of conversation protocols wherein the most important aspect of a conversation protocol is not the set of communicative actions involved in that protocol but the effects or the states that these actions bring about. The basic idea is that since protocols are used to do certain tasks or to bring about certain state of affairs in the world, one should identify the important landmarks or state of affairs that are brought about by and during the execution of a protocol. Conversation protocols can then be expressed at an abstract level as partially ordered landmarks where each landmark is characterized by the propositions that are true in the state represented by that landmark. The partially ordered landmarks represent a family of protocols. Communicative actions are, then, the tools to realize concrete protocols from a landmark-based representation. Besides contributing to formal analyses of protocol families, the landmark-based representation facilitates dynamically choosing the most appropriate action to use next in a conversation, allows compact handling of protocol exceptions, and in some cases, even allows short-circuiting a protocol *execution* by opportunistically skipping some intermediate landmarks.

Despite various research on conversation protocols, there was still a need for a proper formalism for protocols that is suitable for automated reasoning. Based on the definition of conversation protocols as a pattern of communicative actions, we suggested a formalism to address this concern. In this formalism, concrete protocols along with their precondition and goal as action expressions using dynamic logic constructs, thus opening the possibility of applying existing formal theories of dialogue and teamwork, such as joint intention theory, to protocols represented as joint action expressions. In fact, this is how we plan to use conversation protocols within STAPLE. We represent them as action expressions that would get executed automatically without having to implement a separate dialogue manager. The details of our CoABS research on conversation protocols appear in publications listed below.

2.4 Conclusion

We wrote and submitted a journal paper on STAPLE for the "Annals of Mathematics and Artificial Intelligence" Journal special issue on "Logic-based agent implementations". A draft of that paper is attached in the appendix. As for the current state of STAPLE, there have been few bug fixes and enhancements beyond what is described in the paper. STAPLE agents currently reason about semantics of communication using first principles and we expect to demonstrate that it can execute conversation protocols expressed as joint action expressions transparently. About 4-6 weeks of programming is needed to get the STAPLE interpreter into a stable condition, and to get the teamwork based demos working properly. We expect a first public version of STAPLE to be released to the research community by end of the current year.

2.5 Publications:

Kumar, Sanjeev, Huber, Marcus J., and Cohen, Philip R. Representing and Executing Protocols as Joint Actions. In *Proceedings of the First International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS 2002)*, ACM Press, Bologna, Italy, July 15-19, 2002

Kumar, Sanjeev, Huber, Marcus J., and Cohen, Philip R. Direct Execution of Team Specifications in STAPLE (Poster Summary). In *Proceedings of the First International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS 2002)*, ACM Press, Bologna, Italy, July 15-19, 2002.

Kumar, S.; Huber, M. J.; Cohen, P. R.; and McGee, D. R. Toward a Formalism for Conversation Protocols Using Joint Intention Theory. *Computational Intelligence Journal (Special Issue on Agent Communication Language)*, Brahim Chaib-draa and Frank Dignum (Guest Editors), Vol. 18, No. 2, pages 174-228, 2002.

Huber, Marcus J., Kumar, Sanjeev, Cohen, Philip R., and McGee, David R. A Formal Semantics for Proxy Communicative Acts. In *Proceedings of the Eighth International Workshop on Agent Theories, Architectures, and Languages (ATAL-2001)*, Seattle, Washington, USA, August 1-3, 2001.

Kumar, Sanjeev, Huber, Marcus J., McGee, David R., Cohen, Philip R., and Hector J. Levesque. Semantics of Agent Communication Languages for Group Interaction. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI'00)*, American Association for Artificial Intelligence Press, Austin, Texas, July 30-August 3, 2000, pages 42-47.

3 MULTIMODAL INTERACTION IN FIELD AND MOBILE ENVIRONMENTS

3.1 Summary

Multimodal interaction allows users to engage computer systems using the best combination of modalities for the situation and task. Research has documented that because multimodal interaction merges information from multiple sources, it can overcome errors in the individual input modalities, thereby leading to more robust performance. This process, known as mutual disambiguation (MD) [7], has been shown to lead to relative error rate reductions in speech recognition of 19-45% in challenging environments, such as with accented speakers, moderate noise, and moderate motion [3, 8]. The purpose of this project was to test the hypothesis that multimodal interaction would lead to improved performance over the individual modalities in a challenging environment in which subjects were highly exerted, breathing heavily, and physically tired.

We conducted a study in which users engaged in strenuous activity while having to perform map-based computer tasks using portable computing equipment. Specifically, while engaged in an "anti-kidnapping" scenario, users ran across an uneven field, carrying a PDA and a baby carrier, while competing against time. At each "station", the user performed a series of multimodal tasks using the PDA, and then ran to another location. This activity made the users breathe heavily (thus interfering with the speech recognition), and become tired and fatigued (interfering with both the speech and gesture recognition) during the course of the study. The environmental noise included loud ambient noise, such as that generated from a central-cooling system, as well as various kinds of unpredictable ones, such as noise from vehicles or a helicopter, lawn mowers being used in the vicinity, passers-by chatting loudly or talking on cell-phones, and noise from people suddenly arriving to play a game.

The study had three phases viz. stationary, running, and running with a baby carrier. The overall multimodal success rate was about 78%, the speech recognition rate was 75.4%, and the gesture recognition rate was 92.8%. The mutual disambiguation (MD) rate was 12.8%, meaning in 12.8% of the successful commands, either speech or gesture's top candidate hypothesis was incorrect. The MD rate increased with exertion, and the MD rate in the most exerted condition was twice that in the control condition (8.3% in the control, and 14.8% and 16.5% respectively in the two running phases), thus providing evidence of robustness of multimodal interaction in exerted field and mobile environments. The overall performance of the system remained stable, despite the increased exertion. As expected, performance of the gesture recognizer degraded

from the control to the exerted conditions (from 97.6% to 92.1% to 88.9%). However, the most surprising result was that the performance of the speech recognizer was not significantly different in the three conditions (73.5% in the control phase, 75.2% in the running phase, and 76.9% in running with the baby phase).

Overall, our hypothesis that mutual disambiguation would stabilize system performance in the face of exertion was supported. Gesture/sketch recognition was more severely impacted by exertion than was speech, but the multimodal architecture compensated for these errors. However, we are still quite curious about the speech recognition results, which are counterintuitive. One observation that might explain them was that users tended to speak louder when they were exerted. An alternative explanation could be that the speech recognizer running on the PDA performed poorly in those cases in which it did not get the correct recognition —a hypothesis suggested by the fact that the correct utterance was not in the speech n-best list in a large percentage (62%) of the cases that the speech recognizer failed to recognize correctly. This phenomenon needs to be further investigated using the corpus collected during the study. We expect that running the recorded speech through a better recognizer might show the speech recognizer performance to be better for the control, but decrease with increased in exertion, but the system would still have stable performance because of mutual disambiguation.

3.2 The Study

The study involved collecting data on speech, gesture, and multimodal recognition rates under exerted conditions along with a measure of physical exertion to correlate with the performance of the various recognizers. The subjects interacted with the Quickset multimodal system running on HP iPaq PDAs in a real field setting. We developed an anti-kidnapping scenario that required subjects to run as fast as they could between each interaction with the system. The participants were outfitted with two close-talking microphones, a cap to secure the microphones, a voice recorder, a heart-rate transmitter on their chest and a watch on their arm to record the heart-rate data, and a small fanny-pack to hold the voice recorder. The speech recognition was done on the PDA itself but the gesture and multimodal recognition was done on a server machine connected to the PDA over a wireless network.

3.2.1 Storyline

The subjects were told that their task was to rescue kidnapped baby Jessie. The police psychological profilers believe that a psychopath kidnapped her because various objects arranged in strange shapes were left at the scene. Jessie's older sister Mary saw the kidnapping happen at a distance and decided to follow the kidnapper. At each object left by the kidnapper, she left an object with a pointing aspect to it to point to the next place she saw the kidnapper put the object, and hence the subject could follow the arrows to where the baby might be found.

The first priority of the subjects is to retrieve the kidnapped baby. Another high priority is to find and arrest the kidnapper, and for this reason, the subjects need to send information about clues from the field to the police headquarters for further analysis. The subjects would need to follow the clues (shapes and arrows) in order, and enter the information into the PDA. This data gets beamed back to the headquarters so that the detectives and criminal psychologists can puzzle out the kidnapper and get his motive and possibly his location. At the end of the trail of clues, the subject would find baby Jessie abandoned by the psychopath. The subjects need to pick up Jessie and follow the clues back to the starting point, at each point again entering the information in the PDA so as to double check them at the headquarters.

3.2.2 Equipment and Infrastructure

The equipment used in the study consisted of two Compaq iPaq PDAs, one with a 206 MHz Intel StrongArm processor running Pocket PC 2000, and the other with a 400 MHz Intel XScale processor running PocketPC 2002. Each PDA communicated with server programs over a wireless 802.11b network. To provide the network, we installed a mobile antenna for one end of the course, and a roof-top wireless antenna at the other end. One iPAQ had a built-in wireless card while the other iPaq used an Orinoco PCMCIA wireless card connected to a 2db external antenna. This 2db antenna was mounted in a backpack that was carried by the subjects in the first part of the study (1-PDA setup) when the subjects ran with the PDA, and was carried by the research assistant during the second part of the study (2-PDA setup).

For interacting with the machines, we employed two close-talking microphones, and a digital voice recorder. One of the iPaqs had a built-in external microphone jack, and the other iPaq had to be modified to use its headphone jack as an external microphone jack. One of the close talking microphones was attached to the PDA and was used for speech recognition by the Quickset multimodal system. The other microphone was attached to the voice recorder to provide data for scoring the speech recognizer, for calculating the respiration rate as a measure of exertion, and for using that data offline to evaluate other speech recognizers for use in field and mobile settings. During the 2-PDA setup, the subject would run to a station, take the PDA from the research assistant, plug in one of the microphones, enter data, unplug the microphone and return the PDA to the research assistant before running to the next station.

In order to gather heart-rate data, a Polar heart-rate transmitter, and a Polar heart-rate recording watch were used. In order to provide a consistent signal, the heart-rate transmitter was smeared with ECG conducting gel before it was put on the subjects. The time on the heart rate monitor was synchronized with the server machine used for the study before each subject began. It was set to record heart rates in beats per minute every 5 seconds. This heart rate data was then uploaded to a database using an infrared connection to the analysis machine, and is the primary measure of exertion in this study.

Each PDA was running the Quickset user interface integrated with ScanSoft's ASR3200 speech recognition engine, as well as their TTS3000 text-to-speech engine. The speech recognition was done locally on the PDA while the ink was shipped over the wireless network to the backend servers for gesture recognition, and for multimodal integration with the output of the speech recognizer. The resulting multimodal command was sent back to the PDA over the same wireless network for display on a map of the campus by the Quickset user interface.

One of the major issues that we became aware of during the pilot stages of the study was that the standard 802.11b wireless network is unable to keep up while the user was running with the PDA. In this configuration, our Quickset software became disconnected from the network frequently. It typically took about 15-20 seconds to connect back to server machines after a disconnection. Because this time lag was clearly not acceptable for the study, we explored several options, such as using different wireless cards, using the external 2db antenna in a backpack. By removing all possible sources of wireless interference, including Bluetooth transmission, using a specialized wireless network id for the study, and fixing the IP address of the PDA instead of using DHCP, we were finally able to get the system to work reasonably well. Unfortunately, we were still unable to completely prevent the disconnections from occurring. This led to the 2-PDA setup using one stationary PDA at each end of the course. The output of the speech, gesture, and multimodal recognizers were recorded at the backend for data analysis.

The baby carried by the subjects during the second running phase was a 48cm Zapf Creations ChouChou brand, and the carrier was an Evenflo PortAbout with removable base. The carrier, and the baby together weighed roughly 7 lbs.

Figure 5: Subject in experimental gear

The picture in Figure 5 shows a test subject geared up during pilot testing of the experimental setup. One can see the two microphones, the baby in the carrier, the PDA, and wireless antenna in the backpack. Figure 6 shows the pilot subject interacting with the PDA at one of the stations.

Figure 6: Subject at a station

3.2.3 Subjects & Multimodal Task

A total of 25 paid volunteers participated in the study. All subjects were male native speakers of American English, between 20 and 49 years of age, with varying levels of cardiovascular fitness. Previous studies have shown that the performance of speech recognizers is degraded for non-native speakers of English resulting in higher mutual disambiguation for such speakers. Consequently, only native speakers of American English were used in the present study so as to minimize sources of errors in speech recognition other than those caused by exertion and environmental conditions. The reason for using only male subjects was the observation during pilot testing that this particular speech recognizer on the PDA performed sufficiently poorly for female speakers even under normal conditions that it was not usable.

Figure 7: Heart of marbles and knife

The task involved entering two kinds of entities into the PDA using speech and gesture. One of the entities was a "shape" formed by arranging smaller objects such as marbles or pencils. The other entity was an object with an inherent pointing aspect that we refer to as the "arrow". For example, Figure 7 shows a heart shape made out of marbles, and a knife pointing towards another location. Similarly, Figure 8 shows another entity pair (a triangle of cigarettes, and a screwdriver pointing in a certain direction).

Figure 8: Triangle of cigarettes and screwdriver

For each "shape" the subjects were asked to draw the shape, such as a heart or a triangle, on the PDA using the stylus and say what object the shape was made of (such as "marbles" or "cigarettes"). For the "arrow" object, they were asked to draw an arrow on the PDA while saying, "[name of object] pointing this/that way". For example, for Figure 7, the subject would draw an arrow and speak "knife pointing this way", and for Figure 8, the subjects would draw an arrow and say "screwdriver that way". Subjects were told to attempt no more than three times to enter the data at each station before moving on to the next.

3.2.4 Design of the experiment

A within-subject design was used for the study. The three conditions compared were the control (or stationary) condition, the running condition, and the running with a load condition, in that order. On an average, the subjects spent approximately 45–60 minutes during the actual study (not counting the time for instructions and practice). We did not counterbalance the order of trials because of the unpredictability in the time that would have been needed for subjects' heart rate, respiration, and limb fatigue to return to the control level after exertion.

The subjects were first given instructions inside the lab and allowed to become familiar with the system through practice until they felt comfortable using it. The subjects' heart rate was recorded during this practice session in order to obtain a resting level. Thereafter, the subjects were equipped with the rest of the experimental gear and taken outdoors for the actual study. As described earlier, the storyline for the study involved a kidnapped-child rescue situation in order to simulate the conditions under which rescue workers and military personnel may be operating. This scenario involved saving a kidnapped infant which had to be done as fast as possible, thus adding an element of time pressure. In addition, the subjects were told that the participant who completed the study in shortest time would be given an additional $100. The reason behind this reward was to stimulate a race against time that would get subjects to exert themselves as much as they could, thus allowing us to see the effect of exertion on the performance of speech, gesture, and multimodal recognition. The study was carried out over the period of more than a month under weather conditions that varied from rainy and windy to bright and sunny. It was assumed that the effect of unpredictable random outdoor noise would average out over the period of the study both within and among subjects. Therefore, exertion would be the dominant factor influencing all types of recognition. For safety reasons, the heart rate monitor was set up to beep at a maximum heart-level calculated per subject depending on his age, and the experiment would have been discontinued if a subject got over-exerted (as indicated by beeping of the heart rate monitor).

3.3 Procedure

The subjects first completed ten pairs of multimodal commands while standing in the field where the study was set up. The subjects were shown a set of ten photographs of hypothetical pairings of "shapes" and "arrows" that they had to enter into the PDA. Most of these combinations of shapes and arrows were not used in the running conditions. This part of the experiment served as the control phase as the subjects were not yet exerted but had enough practice using the PDA, and we can assume that the external ambient noise conditions (other than the occasional, unpredictable sounds) remained the same during this control phase and the subsequent running phases.

There were ten stations, each of which was set up with a "shape made of objects" and an "arrow. The stations were numbered from one to ten and the "arrow" at each location, except for location ten, pointed to the next location. The arrow at location ten pointed to the location of the baby. The stations were arranged diagonally along the arcs of a circle in the Oregon Graduate Institute (OGI) campus. Each station was marked with a numbered flag that the participants could see when they got close enough. This allowed the subjects to run directly towards the next station without wasting time or having too much opportunity to lower their heart rate

and respiration rate. The overall distance run by the participants was 1.6 km, with an average of 90 m between each station.

After the control phase, the subjects proceeded to the first station to begin the first running condition. At each station, the subjects would enter into the PDA the "shape" and the "arrow" that they found at that station, and then run in the direction of the arrow at that station to reach the next station. This was the first running phase of the experiment. After completing ten pairs of commands in this condition (one pair at each station), the subjects would retrieve the baby in the baby carrier and returned to location ten. Subjects were offered cold water or sports drink at this point to prevent them from getting dehydrated. Thereafter, the subjects began running their way back through all the ten stations in reverse order from station ten to station one, and re-entering information at each station. However, this time they were required to carry the baby in a baby carrier in their writing hand. This was the second running phase of the experiment and was designed to test the effect of arm fatigue on gesture recognition when interacting with the PDA using stylus. After finishing the course, participants returned to the lab for a short interview.

The study had two parts. The first 8 subjects carried the PDA along with them when they ran, and the remaining 17 subjects had a 2-PDA setup where the PDAs were stationary (held by research assistants) at each end of the course. During the second (2-PDA) part of the study, there was a research assistant at each station holding the PDA for the subject, and hence the subjects knew exactly where to run. The ground was uneven and at times slippery and subjects had to be careful while running thus adding to their cognitive load.

3.4 Measures

Three kinds of data were captured per subject: speech was captured by a digital voice recorder, heart rate data gathered by a heart rate recorder, and a log file that recorded the outputs of speech, gesture, and multimodal recognitions along with their timestamps. The voice recording was converted into a "wave" file, transcribed for spoken utterances as well as for exhalations (to calculate respiration rates) using a signal analysis tool called PRAAT. The transcribed speech and the heart rate data along with their timestamps were uploaded into an Access database using Python scripts. Similarly, the log files were parsed and analyzed using Python scripts. The relevant information for scoring the performance of speech, gesture, and multimodal recognition was uploaded into the same Access database along with their timestamps and linked to the transcribed speech and heart rate data. A set of GUI tools were developed using Python and Java to help a researcher score the database.

One of the annotations used in the scoring was to mark whether or not the output of a recognizer was functionally correct. There are several instances when a "close-enough" item on the speech or the gesture n-best list results in the correct multimodal command being on the top of the MM n-best list. In this sense, the incorrect but "close-enough" item can be regarded as being functionally correct even though it is lexically incorrect with respect to the subject's actual speech (or gesture). Just as with lexically correct items, there may be mutual disambiguation if the functionally correct item is pulled up from its position in its n-best list by another modality. Our primary interest for the current study is about lexical correctness, though both functional and lexical correctness were analyzed.

MD occurs when the top-ranked multimodal command includes an interpretation from speech and/or from gesture that is not itself top-ranked for that modality. The MD *rate* is thus calculated as the percentage of *correct* multimodal commands that had mutual disambiguation between the input modalities. This definition of MD rate is different from that in the literature (Oviatt, 1999) because we are only interested in the contribution of MD towards correct multimodal recognition. For the purpose of calculating MD, all items on the speech n-

best list whose difference in probability was less than 0.0001 were counted as having the same probability, and hence having the same rank.

As mentioned earlier, data was collected for 25 subjects out of which the first 8 subjects used the 1-PDA setup (they carried the PDA with them as they ran) and the other 17 subjects used the 2-PDA setup (two stationary PDA's were held by research assistants at the two end of the course). Because the system performance from the 1-PDA condition was frequently influenced by wireless network problems and/or slow response times, it was decided to score and analyze the data for the 2-PDA setup only at this time. Out of the 17 subjects with 2-PDA setup, 1 subject could not complete the course, and the data for two of the subjects was badly out of synch and hence could not be scored. Out of the remaining 14 subjects, one subject made too many user errors (such as drawing line instead of drawing an arrow). We therefore analyzed data from 13 subjects. Also, although the analyses were done twice, once counting only the first multimodal attempts, and then counting all multimodal commands. We present only the results for the first attempts.

3.5 Results

The results that follow are from the first attempts analysis of 13 subjects (from the 2-PDA setup) counting only the lexically correct recognitions as correct recognition results. The annotation "L" in the various figures indicates lexical correctness.

3.5.1 Summary

The variation of multimodal success rate (MMS), speech recognition rate (SR), and gesture recognition rate (GR) across the three study conditions is given in Figure 9.

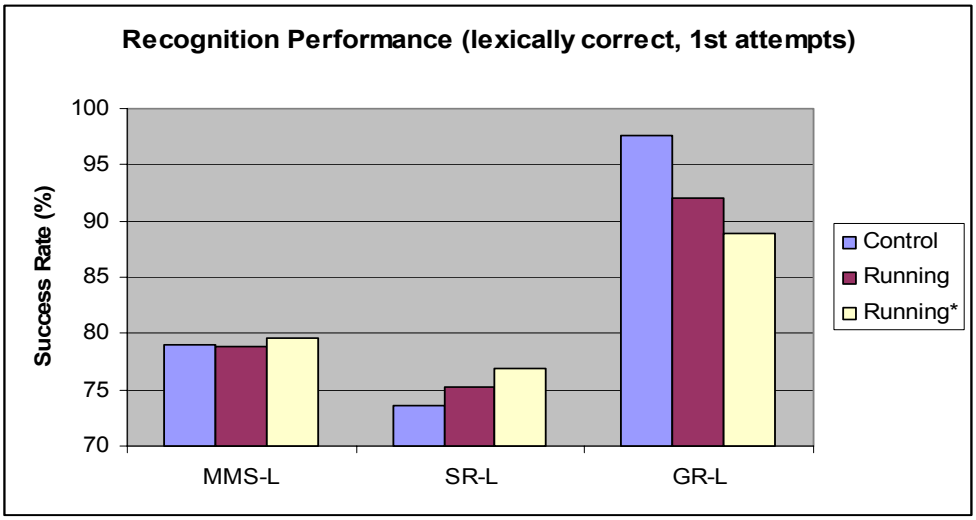

Figure 9: Performance of multimodal, speech, gesture recognition

This figure shows that the performance of the gesture recognizer decreases from the control to the two running cases, though the decrease from the first running condition to the second running condition (running*) is not statistically significant. Contrary to expectations, this figure does not show the performance of the speech recognizer to be affected by running. The multimodal success rate increases very little from the control to the running cases, and this increase, however small, may be attributed to the better speech recognition. Figure 10 shows variation of mutual disambiguation rate (MD) across the three conditions - control, running, and running

with baby (running*). This figure illustrates that the MD rate in the two running conditions is nearly twice that of the MD rate in the stationary (control) condition.

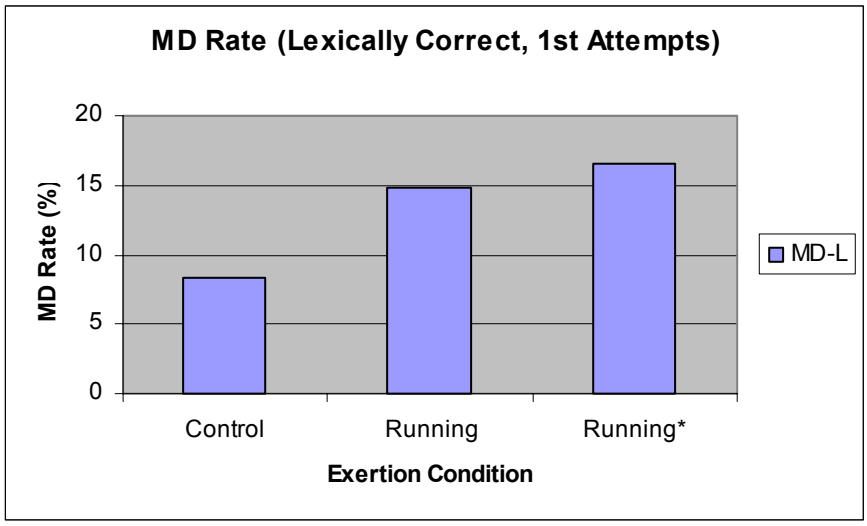

Figure 10: MD rate across conditions

The fact that the contribution of MD increases from the control to the running conditions even when the performance of the gesture recognizer is degraded shows that (1) the speech is pulling up the gesture most of the time, and (2) without mutual disambiguation, the performance of the system would have been degraded significantly in the running conditions as compared with the stationary condition.

3.5.2 Tests for statistical significance

Next, we statistically analyze these measures across the three conditions in order to determine which of the observed results are statistically significant. The fact that each subject went through all the three conditions allows us to compare each measure (MD, SR, or GR) for every subject between two conditions at a time (between control and running, between control and running*, and between running and running*) using a pair-wise t-test. All t-tests were one-tailed, α was 0.05 and the null hypothesis was that the difference of means is zero.

Figures 11, 12, and 13 summarize the p-values for the difference in means between various pairs of conditions for 13 subjects for the lexically correct case using 1-tailed pair wise t-test for MD, gesture recognition and speech recognition rates respectively.

	Running	**Running with baby**
Control	0.038827	0.013211
Running		0.301373

Figure 11: P values for pair-wise t-tests on MD rate (13 subjects, lexically correct)

The low p-value for the control vs. running, and control vs. running with baby conditions in Figure 11 confirms our hypothesis that there is statistically significant increase in the MD rate between the control and the two running conditions. However, the high p-value for the two running conditions show that one cannot confirm whether or not there is any statistically significant difference in the mean MD rates for those two conditions.

	Running	Running with baby
Control	0.013068	0.002410
Running		0.106856

Figure 12: P values for pair-wise t-tests on gesture recognition (13 subjects, lexically correct)

The low p-value for the control vs. running, and control vs. running with baby conditions in the Figure 12 confirms our hypothesis that there is statistically significant decrease in the performance of gesture recognizer between the control and the two running conditions. However, the high p-value for the two running conditions show that one cannot confirm whether or not there is any statistically significant difference in the mean gesture recognition rates for those two conditions.

	Running	Running with baby
Control	0.379138	0.298350
Running		0.315661

Figure 13: P values for pair-wise t-tests on speech recognition (13 subjects, lexically correct)

The high p-values in all cases in Figure 13 says that we cannot demonstrate a difference in performance of speech recognition from the control to the two running conditions, or between the two running condition.

3.5.3 Correlation with heart rate

One of the goals of the current study was to find how the speech, gesture, and multimodal recognitions perform under exerted conditions. We use heart rate as the primary measure of exertion, other measures being respiration rate, and running speed. As mentioned earlier, each subject's heart rate was recorded every 5 seconds during practice session (to get resting heart rate) and also during the actual study outdoors. The absolute heart rate of each subject varied widely, and the absolute heart rate data from different subjects could not be combined. Therefore, each absolute heart rate data was converted into relative heart rate as a percentage of that user's resting heart rate. The success and failure of each speech, gesture, and multimodal command was then correlated with both the absolute and relative heart rates for all subjects. The overall correlation for all subjects combined was computed by combining the success and failure data as well as the relative heart rates for all subjects. Figure 14 presents the correlation of various success rates with relative heart rate for all 13 subjects combined. Surprisingly, there is negative correlation only between gesture recognition and the relative heart rate.

	Correlation with Relative Heart Rate
Multimodal Success Rate	0.057801
Speech Recognition Rate	0.086210
Gesture Recognition Rate	-0.169875

Figure 14: Correlation with relative heart (13 subjects, lexically correct)

3.6 Conclusions

The study demonstrated that multimodal interaction with map-based PDAs that offer mutual disambiguation of modalities can support stable performance when subjects are highly exerted. The QuickSet multimodal system showed a doubling of its mutual disambiguation rate (8% to 16%) from stationary to exerted conditions, while

the overall multimodal success rate stayed constant. Exertion was found to affect gesture/sketch recognition more than it did speech recognition, with the gesture recognition rate being significantly lower during the exerted condition than in the control, while speech remained stable. However, follow-up research needs to investigate whether a better speech recognition system might have given still better results in the control condition, which might then result in a decrease when exerted.

3.6.1 Equipment observations

The study was originally designed and pilot tested using a single PDA that was wirelessly connected to the campus computer network using 802.11b. However, we found that various 802.11b receivers (especially the one on the HP iPAQ) in the PDAs could not properly maintain network connections while the subject was running, and that switching of network base stations as the subject ran would often cause a substantial time delay in reacquiring the network, often leading to poor multimodal performance. This problem led us to abandon the original method of having subjects carry the PDA from station to station, and instead run from one station to another where they picked up a PDA that was already on the network.

The Quickset system was also enhanced with an ability to communicate with an Emtac Bluetooth GPS receiver, thus enabling the system to display the position, orientation and speed of a subject on the PDA in real time. The objective was to provide the subjects with a sense of their location and orientation with respect to the study area, and the running speed was to be used as a third measure of exertion. However, it was found during the pilot studies that there were too many problems with the wireless network during the running conditions. Moreover, the Bluetooth GPS unit and the transmission of GPS data from the PDA were slowing down the system's response. Because that response time needed to be kept within 3-4 seconds in order not to allow enough time for the heart rate of the subjects to come down between commands, it was decided not to use the GPS for the study. Instead we used a rough estimate of speed calculated from the known distance between each station, and the time recorded by the system between the subject's last command at the last station, and the first command at the current station. Finally, during pilot testing, the equipment setup employed a Plantronics' Bluetooth wireless microphone to communicate with the PDA. However, the design of the microphone (which wrapped around the ear) was too unstable to keep positioned correctly when worn by a running subject. In addition, the battery life was too short to last for an entire session. It was therefore decided to go with wired microphones instead for the actual study.

3.6.2 Future data analysis

The data collected by the current study offers several opportunities for further analysis and research.

(1) The respiration needs to be scored for each subject and correlated with the speech, gesture, and multimodal success rates.

(2) The running speed of subjects between each station needs to be estimated and correlated with the various success rates. This will provide a better measure of exertion, for example, some subjects were so tired by the third segment that they merely walked rather than ran. Their heart rate and respiration would not then necessarily be an indicator of their exertion level.

(3) In order to further investigate the counterintuitive finding that speech recognition was unaffected by exertion, the recorded voice should be run through another recognizer, or through the same recognizer with different thresholds, and the analyses redone. When the current recognizer failed, most of the time (62%) the correct item was not among its hypotheses (i.e., in its n-best list). By comparing with a different recognizer, or with different n-best thresholds, we would hope that the correct item would be somewhere in the n-best list, which is required if mutual disambiguation is to be of any help. However, a longer n-best list comes at a cost of considerable computational resource and thus, there is always a compromise between the system response time and its accuracy.

4 BIBLIOGRAPHY

[1] Cohen, P. R., Cheyer, A., Wang, M., and Baeg, S. C. An open agent architecture. In *Proceedings of AAAI Spring Symposium: Software Agents*, Menlo Park, CA, 1-8, 1994.

[2] Cohen, P. R. and Levesque, H. J. Teamwork. *Nous*, 25(4): 487-512, 1991.

[3] Kaiser, E., Olwal, A., McGee, D. R., Benko, H., Corradini, A., Li, X., Cohen, P. R., and Feiner, S. Mutual Disambiguation of 3D Multimodal Interaction in Augmented and Virtual Reality. In *Proceedings of 5th International Conference on Multimodal Interfaces*, Vancouver, BC, 2003.

[4] Kumar, S. and Cohen, P. R. Towards a Fault-Tolerant Multi-Agent System Architecture. In *Proceedings of Fourth International Conference on Autonomous Agents (Agents 2000)*, Barcelona, Spain, ACM Press, 459-466, 2000.

[5] Kumar, S., Cohen, P. R., and Levesque, H. J. The Adaptive Agent Architecture: Achieving Fault-Tolerance Using Persistent Broker Teams. In *Proceedings of Fourth International Conference on Multi-Agent Systems (ICMAS 2000)*, Boston, MA, USA, IEEE Press, 159-166, 2000.

[6] Martin, D., Cheyer, A., and Moran, D. The Open Agent Architecture: A framework for building distributed software systems. *Applied Artificial Intelligence: An International Journal*, 13(1-2), 1999.

[7] Oviatt, S. L. Mutual disambiguation of recognition errors in a multimodal architecture. In *Proceedings of Conference on Human Factors in Computing Systems: CHI '99*, New York, N.Y, ACM Press, 576-583, 1999.

[8] Oviatt, S. L. Multimodal System Processing in Mobile Environments. In *Proceedings of Thirteenth Annual ACM Symposium on User Interface Software Technology (UIST'2000)*, New York, ACM Press, 21-30, 2000.